Estimating Planetary Habitability via Particle Swarm Optimization of CES Production Functions.

Abhijit J. Theophilus

January 14, 2018

1 Introduction

The search for extra-terrestial life and potentially habitable extrasolar planets has been an international venture demanding large investments in cost and effort, since Frank Drake's attempt with Project Ozma in the mid-20th century. The first exoplanet was officially confirmed in 1992 which marked the start of a trend that has lasted 25 years and yielded over 3,700 confirmed exoplanets. There have been attempts to assess the habitability of these planets and to assign a score based on their similarity to Earth. Two such habitability scores are the Cobb-Douglas Habitability Score (CDHS) and the Constant Elasticity Earth Similarity Approach (CEESA) score. Estimating these scores involves maximizing a production function subjected to a set of constraints on its input variables.

Under most paradigms, maximizing a continuous function requires calculating a gradient. This may not always be feasible for non-polynomial functions in high-dimensional search spaces. Further, subjecting the input variables to constraints, as needed by CDHS and CEESA, are not always be straightforward to represent within the model. This paper presents an approach to Constrained Optimzation (CO) using the swarm intelligence metaheuristic. Particle Swarm Optimization (PSO) is a method for optimizing a continuous function that does away with the need for a gradient. It employs a large number of particles that traverse the search space converging toward a global best solution encountered by at least one of the particles.

Particle Swarm Optimization is a distributed method that requires simple mathematical operators and short segments of code, making it a lucrative solution where computational resources are at a premium. Its implementation is highly parallelizable. It scales with the dimensionality of the search space. The standard PSO algorithm does not deal with constraints, but through variations in initializing and updating particles, constraints are straightforward to represent and adhere to, as seen in Section 3.2.

PSO has been adapted to a wide range of design optimization problems, e.g., networks and VLSI design. It has found applications in machine learning under clustering, feature detection and classification. As a modeling paradigm, it has been used for constructing customer satisfaction models, modeling MIDI music and chaotic time series modeling.

This paper demonstrates the applicability of Particle Swarm Optimization in estimating CDHS and CEESA scores of an exoplanet by maximizing their respective production functions, discussed in Sections 2.1 and 2.2. CDHS considers the planet's Radius, Mass, Escape Velocity and Surface Temperature, while CEESA includes a fifth parameter, the Orbital Eccentricity of the planet. The Exoplanets Catalog hosted by the Planetary Habitability Laboratory, UPR Arecibo records these parameters for each exoplanet in Earth Units. Section **??** reports the performance of PSO and contrasts it against an earlier effort to estimate these scores using Stochastic Gradient Ascent (SGA).

2 Habitability Scores

2.1 Cobb-Douglas Habitability Score

Estimating the Cobb-Douglas Habitability Score (CDHS) requires estimating an interior CDHS ($CDHS_i$) and a surface CDHS ($CDHS_s$) by maximizing the following production functions,

$$Y_i = CDHS_i = R^{\alpha}.D^{\beta} \tag{1a}$$

$$Y_s = CDHS_s = {V_e}^{\gamma}.{T_s}^{\delta} \tag{1b}$$

where, R, D, V_e and T_s are density, radius, escape velocity and surface temperature respectively. α, β, γ and δ are the elasticity coefficients subject to,

$$0 < \alpha, \beta, \gamma, \delta < 1 \tag{2}$$

Equations 1a and 1b are convex when the returns to scale are either constant or decreasing, marked by two constraints on the elasticity coefficients,

$$\alpha + \beta = 1, \quad \gamma + \delta = 1 \qquad \text{for Constant Returns to Scale (CRS),} \tag{3a}$$

$$\alpha + \beta < 1, \quad \gamma + \delta < 1 \qquad \text{for Decreasing Returns to Scale (DRS).} \tag{3b}$$

The final CDHS is the convex combination of the interior and surface CDHS values as given by,

$$Y = w_i.Y_i + w_s.Y_s \tag{4}$$

2.2 Constant Elasticity Earth Similarity Approach

The Constant Elasticity Earth Similarity Approach (CEESA) uses the following production function to estimate the habitability score of an exoplanet,

$$Y = (r.R^{\rho} + d.D^{\rho} + t.{T_s}^{\rho} + v.{V_e}^{\rho} + e.E^{\rho})^{\eta/\rho} \tag{5}$$

where, E is the fifth parameter denoting Orbital Eccentricity. The value of ρ lies within $0 < \rho \leq 1$. The coefficients (r, d, t, v and e) are constrained by,

$$0 < r, d, t, v, e < 1 \tag{6a}$$

$$r + d + t + v + e = 1 \tag{6b}$$

The value of η is constrained by the scale of production used,

$$0 < \eta < 1 \qquad \text{for Constant Returns to Scale (CRS),} \tag{7a}$$

$$\eta = 1 \qquad \text{for Decreasing Returns to Scale (DRS).} \tag{7b}$$

3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a biologically inspired metaheuristic for finding the global minima of a function. Traditionally designed for unconstrained inputs, it works by iteratively converging a population of randomly initialized solutions, called particles, toward a globally optimal solution. Each particle in the population keeps track of its current position and the best solution it has encountered so far, called *pbest*. Each particle also has an associated randomized velocity used to traverse the search space. The swarm keeps track of the overall best solution, called *gbest*. Each iteration of the swarm updates the velocity of the particle towards its *pbest* and the *gbest* values.

3.1 PSO for Unconstrained Optimation

Let $f(x)$ be the function to be minimized, where x is a d-dimensional vector. $f(x)$ is also called the fitness function. Algorithm 1 outlines the approach to minimizing $f(x)$ using PSO. A set of particles are randomly initialized with a position and a velocity, where l and u are the lower and upper boundaries of the search space. The position of the particle corresponds to its associated solution. The algorithm initializes each particle's *pbest* to its initial position. The *pbest* position that corresponds to the minimum fitness is selected to be the *gbest* position of the swarm.

On each iteration, the algorithm updates the velocity and position of each particle. For each particle, it picks two random numbers u_g, u_p from a uniform distribution, $U(0, 1)$ and updates the particle velocity as indicated in line 11. Here, μ is the friction coefficient and λ_g, λ_p are the global and particle learning rates. If the new position of the particle corresponds to a better fit than its *pbest*, the algorithm updates *pbest* to the new position. Once the algorithm has updated all particles, it updates *gbest* to the new overall best position. A suitable termination criteria for the swarm, under convex optimization, is when the *gbest* position has not changed by the end of the iteration.

Procedure 1 Algorithm for PSO.

Input: $f(x)$, the function to minimize.
Output: global minimum of $f(x)$.

1: **for** each particle $i \leftarrow 1, n$ **do**
2: $p_i \sim U(l, u)$
3: $v_i \sim U(-|u - l|, |u - l|)$
4: $pbest_i \leftarrow p_i$
5: **end for**
6: $gbest \leftarrow \arg\min_{pbest_i, i \in 1,n} f(pbest_i)$
7: **repeat**
8: $oldbest \leftarrow gbest$
9: **for** each particle $i \leftarrow 1, n$ **do**
10: $u_p, u_g \sim U(0, 1)$
11: $v_i \leftarrow \mu.v_i + \lambda_g.u_g.(gbest - p_i) + \lambda_p.u_p.(pbest_i - p_i)$
12: $p_i \leftarrow p_i + v_i$
13: **if** $f(p_i) < f(pbest_i)$ **then**
14: $pbest_i \leftarrow p_i$
15: **end if**
16: **end for**
17: $gbest \leftarrow \arg\min_{pbest_i, i \in 1,n} f(pbest_i)$
18: **until** $|oldbest - gbest| < threshold$
19: **return** $f(gbest)$

3.2 PSO with Leaders for Constrained Optimization

Although PSO has eliminated the need to estimate the gradient of a function, as seen in Section 3.1, it still is not suitable for constrained optimization. The standard PSO algorithm does not ensure that the initial solutions are feasible, and neither does it guarantee that the individual solutions will converge to a feasible global solution. Solving the initialization problem is straightforward, resample each random solution from the uniform distribution until every initial solution is feasible. To solve the convergence problem, each particle uses another particle's *pbest* value, called *lbest*, instead of its own to update its velocity. Algorithm 2 describes this process.

On each iteration, for each particle, the algorithm first picks two random numbers u_g, u_p as before. It then selects a *pbest* value from all particles in the swarm that is closest to the position of the

particle being updated as its *lbest*. The *lbest* value substitutes $pbest_i$ in the velocity update equation. While updating *pbest* for the particle, the algorithm checks if the current fit is better than *pbest*, and performs the update if the current position satisfies all constraints.

Procedure 2 Algorithm for CO by PSO.

Input: $f(x)$, the function to minimize.
Output: global minimum of $f(x)$.

for each particle $i \leftarrow 1, n$ **do**
  repeat
    $p_i \sim U(l, u)$
  until p_i satisfies all constraints
  $v_i \sim U(-|u-l|, |u-l|)$
  $pbest_i \leftarrow p_i$
end for
$gbest \leftarrow \arg\min_{pbest_i, i \in 1,n} f(pbest_i)$
repeat
  $oldbest \leftarrow gbest$
  for each particle $i \leftarrow 1, n$ **do**
    $u_p, u_g \sim U(0, 1)$
    $lbest \leftarrow \arg\min_{pbest_j, j \in 1,n} \|pbest_j - p_i\|^2$
    $v_i \leftarrow \mu.v_i + \lambda_g.u_g.(gbest - p_i) + \lambda_p.u_p.(lbest - p_i)$
    $p_i \leftarrow p_i + v_i$
    if $f(p_i) < f(pbest_i)$ **and** p_i satisfies all constraints **then**
      $pbest_i \leftarrow p_i$
    end if
  end for
  $gbest \leftarrow \arg\min_{pbest_i, i \in 1,n} f(pbest_i)$
until $|oldbest - gbest| < threshold$
return $f(gbest)$

4 Experiment

Should include,

- Discussing the data set.
- Parameters for the constrained optimization.
- Implementation method.
- Ensuring convergence.

5 Results

Should include,

- The values and proximity to earth's habitability score.
- Values that do not converge. Or were hard to converge.
- Speed of convergence graphs.

- Graph1: Iterations to convergence vs. Number of particles.
- Graph2: Distribution of number of iterations to convergence.
- Graph3: Iterations to convergence vs. Constraint parameters.

6 Conclusions

Should include,

- Why is the speed so important?
- Parallelizable.